Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет по лабораторной работе №4**

**«Избыточное кодирование данных в информационных системах. Код Хемминга»**

Выполнил:

студент 3 курса 1 группы

Парибок И.А.

Минск 2022

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

Задачи:

1. Закрепить теоретические знания по использованию методов помехоустойчивого кодирования для повышения надежности передачи и хранения в памяти компьютера двоичных данных.

2. Разработать приложение для кодирования/декодирования двоичной информации кодом Хемминга с минимальным кодовым расстоянием 3 или 4.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

Весь алгоритм выполнения программы находится в приложении А.

Для начала нужно ввести исходное сообщение, которое будет представлено в двоичном виде.



Рисунок 1 – Исходное сообщение

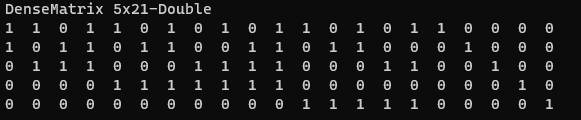


Рисунок 2 – Матрица Хэмминга сообщения

Используя данную матрицу, было вычислено избыточное слово, которое представлено на рисунке 3.



Рисунок 3 – Избыточное слово Xr



Рисунок 4 – Кодовое слово Xn

Для полученного слова Yn = Yk, Yr, используя уже известную проверочную матрицу Хемминга, были вычислены избыточные символы, а также вычислен и проанализирован синдром.

Результат выполнения программы при 0 ошибок представлен на рисунке 5.

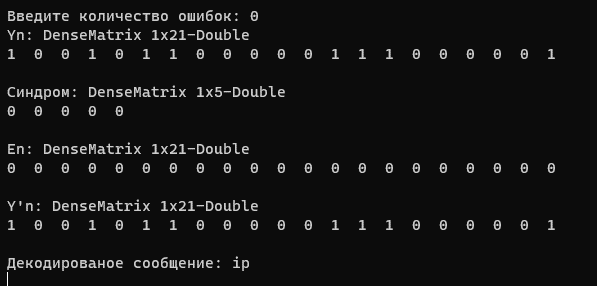


Рис. 5 – Результат выполнения при 0 ошибок

Результат выполнения программы при 1 ошибке представлен на рисунке 6.

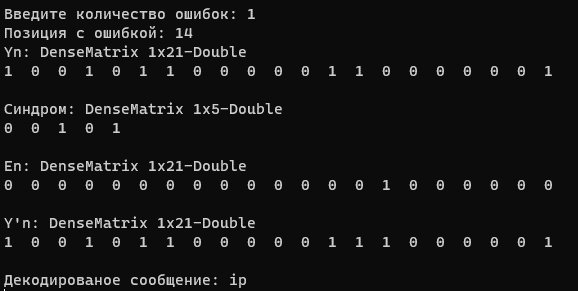


Рис. 6 – Результат выполнения при 1 ошибке

Результат выполнения программы при 2 ошибках представлен на рисунке 7.

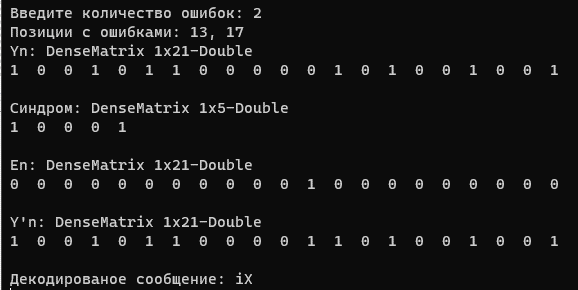


Рис. 7 – Результат выполнения при 2 ошибках

Вывод: в результате данной лабораторной работы было разработано приложение для кодирования/декодирования двоичных данных при использовании кода Хемминга. Проанализировав полученные результаты при декодировании с ошибками, можно заметить, что код Хэмминга успешно работает только при 0 и 1 ошибках и не справляется с 2 ошибками.

**Приложение А**

using System;

using System.Collections;

using System.Text;

using MathNet.Numerics.LinearAlgebra;

using MathNet.Numerics.LinearAlgebra.Double;

namespace Lab3

{

class Program

{

static bool IsPowerOfTwo(ulong x)

{

return (x & (x - 1)) == 0;

}

public static byte[] BitArrayToByteArray(BitArray bits)

{

byte[] ret = new byte[(bits.Length - 1) / 8 + 1];

bits.CopyTo(ret, 0);

return ret;

}

static void Main(string[] args)

{

Matrix<double> Xn, Xr, Xk, Yn, H;

int position\_1, position\_2;

Random rnd = new Random();

Console.Write("Введите строку: ");

string str = Console.ReadLine();

var bytes = Encoding.ASCII.GetBytes(str);

int k = bytes.Length \* 8;

int r = (int)Math.Ceiling(Math.Log(k, 2) + 1);

double[,] g = new double[r, k];

double[,] i = new double[r, r];

for (int ii = 0; ii < r; ii++)

{

i[ii, ii] = 1;

}

int f = 1;

for (int j = 0; j < k; j++)

{

while (IsPowerOfTwo((ulong)f)) f++;

int d = f;

int ff = 0;

while (d > 0)

{

g[ff, j] = d % 2;

d /= 2;

ff++;

}

f++;

}

H = DenseMatrix.OfArray(g).Append(DenseMatrix.OfArray(i));

Console.WriteLine(H);

var bar = new BitArray(bytes);

var xr = new double[r];

for (int ii = 0; ii < r; ii++)

{

for (int j = 0; j < k; j++)

{

xr[ii] += (bar[j] ? 1 : 0) \* H[ii, j];

}

xr[ii] %= 2;

}

Xr = Matrix<double>.Build.Dense(1, r, (ii, jj) => (double)xr[jj]);

Xk = Matrix<double>.Build.Dense(1, bar.Length, (ii, jj) => bar[jj] ? 1 : 0);

Xn = DenseMatrix.OfMatrix(Xk).Append(Xr);

Console.WriteLine($"Xr: {Xr}");

Console.WriteLine($"Xn: {Xn}");

Yn = DenseMatrix.OfMatrix(Xn);

Console.Write("Введите количество ошибок: ");

int errCount = Convert.ToInt32(Console.ReadLine());

if (errCount == 1)

{

position\_1 = rnd.Next(0, Yn.ColumnCount - 1);

Yn[0, position\_1] = (Yn[0, position\_1] + 1) % 2;

Console.WriteLine($"Позиция с ошибкой: {position\_1}");

}

if (errCount == 2)

{

position\_1 = rnd.Next(0, Yn.ColumnCount - 1);

Yn[0, position\_1] = (Yn[0, position\_1] + 1) % 2;

position\_2 = rnd.Next(0, Yn.ColumnCount - 1);

while (position\_1 == position\_2) position\_2 = rnd.Next(0, Yn.ColumnCount - 1);

Yn[0, position\_2] = (Yn[0, position\_2] + 1) % 2;

Console.WriteLine($"Позиции с ошибками: {position\_1}, {position\_2}");

}

if (errCount > 2)

{

Console.WriteLine("Много ошибок");

return;

Console.ReadKey();

}

Console.WriteLine($"Yn: {Yn}");

var S = H \* Yn.Transpose();

bool flag = false;

for (int o = 0; o < S.RowCount; o++)

{

S[o, 0] %= 2;

flag = flag || S[o, 0] != 0;

}

Console.WriteLine($"Синдром: {S.Transpose().ToString()}");

var En = Matrix<double>.Build.Dense(1, H.ColumnCount, 0);

if (flag)

{

for (int o = 0; o < H.ColumnCount; o++)

{

if (H.Column(o).Equals(S.Column(0)))

{

En[0, o] = 1;

}

}

}

Console.WriteLine($"En: {En}");

var Y\_n = DenseMatrix.OfMatrix(Yn) + En;

for (int o = 0; o < Y\_n.ColumnCount; o++)

{

Y\_n[0, o] %= 2;

}

Console.WriteLine($"Y'n: {Y\_n}");

var bitArray = new BitArray(k);

for (int o = 0; o < k; o++)

{

bitArray[o] = Y\_n[0, o] == 1;

}

Console.WriteLine($"Декодированое сообщение: {Encoding.ASCII.GetString(BitArrayToByteArray(bitArray))}");

Console.ReadKey();

}

}

}

Листинг 1 – программа использования кода Хемминга